Visually Exploring Control Flow Graphs to Support Legacy Software Migration

Johannes Bohnet, Jürgen Döllner
Hasso-Plattner-Institut – University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3, 14482 Postdam, Germany
{bohnet, doellner}@hpi.uni-potsdam.de

Abstract: Migrating legacy systems to new platforms represents a frequent challenge to leverage earlier massive capital investments. Prerequisite for performing migration include profound understanding of the system and its components – a cost intensive task if systems and their components are monolithic and highly coupled, and up-to-date documentation and system models do not exist, like in most cases. We propose a software exploration tool that supports migration of legacy systems. It facilitates identification of high-level code components and their interaction in complex legacy systems written in C. Developers can analyze (a) how the system is collaborating with environmental systems and (b) how the core business logic is intertwined with platform-specific code. The tool's key features are (a) extracting control flow graphs and interpreting them within the system's static architecture and (b) providing a visualization front-end for efficient exploration of the analysis results.

We propose a software exploration tool that facilitates the migration process of legacy systems and reengineering tasks, supporting developers during their code analysis tasks. The tool reveals how the legacy system is linked to its environment by showing how the control flow enters and leaves the system during runtime. Additionally, the tool reveals how internal components of the legacy system interact. This way, developers are supported in identifying the boundary between business logic and technical, platform-

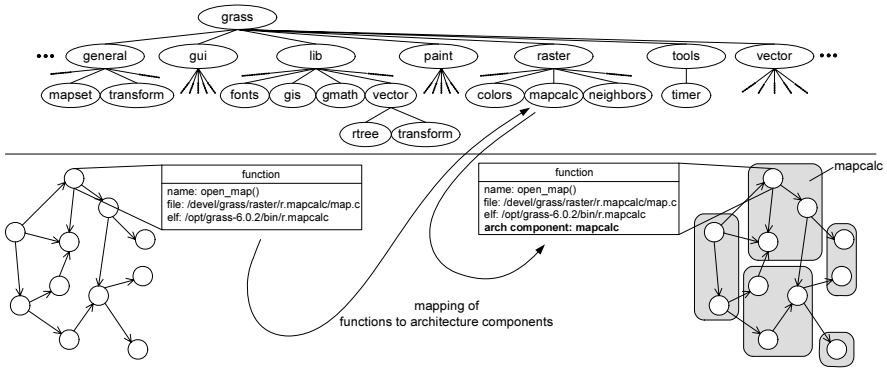

Figure 1: **Top**: The legacy system's reconstructed static architecture. **Bottom**: Control flow graphs created during runtime are mapped onto the system architecture and can be interpreted as high-level interaction between architectural components.

specific code, which should be discarded for a migrated system with high quality. Furthermore, the task of breaking up the monolithic legacy application into a collection of modular components is facilitated as, by visualizing the control flow during runtime, the tool allows developers identifying these components and understanding their interplay. In a sense, the tool facilitates the rediscovery of the system's logical structure. The tool can be applied on complex (>MLOC) software systems written in C. Prerequisites are the availability of the system's source code and an executable binary file that has been build with debug information. Due to these minimal, non-intrusive prerequisites, e.g., no code instrumentation needs to be done, the tool can easily be incorporated into existing migration processes.

The key ideas of the presented approach are illustrated in Figure 1 and 2 showing the process of analyzing the legacy geographic information system *Grass GIS* (http://grass.itc.it):

a) Extracting the legacy system's static architecture by analyzing the source repository structure (Fig. 1 top);
b) Extracting control flows on function level during run-time according to specific system use cases; (Fig. 1 bottom left)
c) Mapping control flows onto the static system architecture – based on functions' debug information. This way, control flow graphs can be interpreted as high-level interaction between architectural components; (Fig. 1 bottom right)
d) Providing an elaborate interactive visualization front-end for exploring both high-level interaction, i.e., interaction of legacy system with its environment and interaction between architectural components of the system, and low-level interaction, i.e., interaction between functions. (Fig. 2)

The tool has been tested on various commercial and open source software systems with implementations ranging from several 100.000 LOC up to several million LOC.

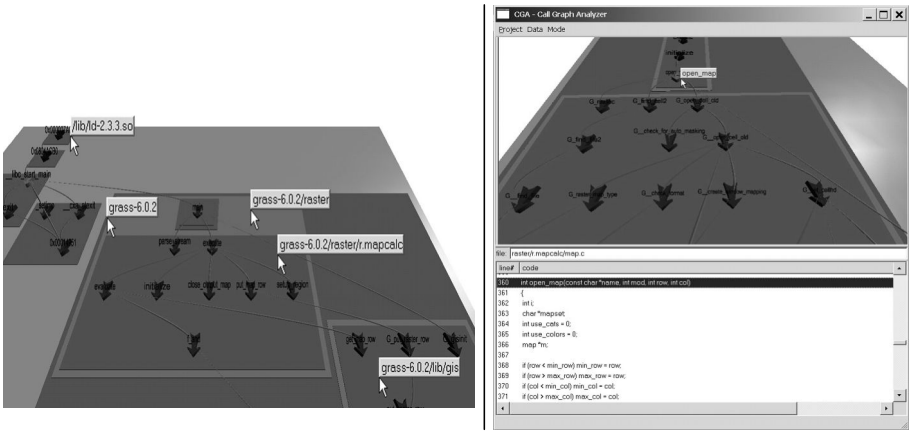

Figure 2: The visualization front-end combines views that permit to interactively explore control flow graphs with views that provide rapid access to corresponding source code representation.